Erledigt macOS High Sierra Erfahrungen

Beitrag von „mhaeuser“ vom 8. Juni 2017, 19:19

> Zitat von kuckkuck
>
> Das Problem ist prinzipiell dass die ded. GPU einen Treiber lädt um Video während des Boots und davor darstellen zu können. In dem Moment wo OS X dann hochgefahren ist und seine Treiber zur Verfügung stellt, ist jedoch bereits ein Treiber geladen und die Karte wird nicht dazu animiert einen neuen, also den von Apple zu laden. Der bereits geladene Treiber ist aber ich Normalfall nicht OS X kompatibel oder nur teilweise. Ein Beispiel für letzteres wären einige R9 Karten, deren Treiber zwar unter OS X zu laufen scheinen, jedoch streiken wenn der Sleep deaktiviert wird, da sie sich eben von Apples Treibern unterscheiden.
> [...]
> 2. die GPU wird kurz bevor Apples Treiber greifen nochmal resettet und nimmt daraufhin auch Apples Treiber an (Beim Boot: GPU eigener Treiber, Nach dem Reset: OS X Treiber)

Sorry, aber das ist viel zu "simplistisch", vor allem für die Grundlage einer technische Recherche.

1) Der Treiber, der beim Bootvorgang geladen wird, bleibt bis kurz nach dem Erscheinen des Apple-Logo aktiv (genaugenommen bis ExitBootServices()), danach schreibt der Kernel direkt in den Framebuffer.
-> Folgen: Der Treiber als solcher "blockiert" einen Apple-Treiber nicht, da er die Kontrolle "zwangsweise" abgeben musste. Dies geschieht aktiv oder passiv (aktiv: falls es einen "Lock" gibt, wird dieser gelöst. passiv: falls es einen "Lock" gibt, wird dieser nicht gelöst. Beispiel: EHCI/XHCI Handoff. In jedem Fall wird das Gerät nicht weiter angesprochen.). Des Weiteren ist er während der Runtime-Phase nicht aktiv und kann daher auch nicht "mit OS X kompatibel sein".

2) Der Treiber, der beim Bootvorgang geladen wird, startet anscheinend (hab' ich nicht verifiziert) den POST und initialisiert die Karte (scheint sie nicht ohne "Anweisung" des Treibers zu tun, da bei dessen Nichtladen ein POST nicht durchgeführt wird).
-> Folgen: Während der Initialisierung werden Register beschrieben. Diese sind i.d.R. viele an der Zahl und können u.U. zu weiteren Änderungen von Registern oder der Funktionsweise der

Karte vornehmen (Analogie: CPU-Modi). Diese können nur durch Kenntnis über die Initialisierung der spezifischen Karte bestimmt und rückgängig gemacht werden. Wenn also niemand Lust auf monatelanges Reverse Engineering auf Quellcode- und Hardware-Ebene hat, sollte man den Gedanken der "manuellen Deinitialisierung" der Karte ganz schnell wieder verwerfen.

3) Das VBIOS ist u.a. eine Initialisierungsroutine. Der Vorgang kann, wie oben beschrieben, nur mit Hardwarekenntnis rückgängig gemacht werden (falls überhaupt möglich). Ein "Neuladen" des VBIOS wird die Änderungen nicht rückgängig machen, da es sich hier nicht um eine OS-ähnliche Umgebung handelt. Die Änderungen sind Hardware-seitig, solange die Karte mit Strom versorgt wird (bzw. die Teile, die die Register etc versorgen).

4) Per UEFI-Spezifikation müssen alle Treiber entladen werden können.
-> Test: Ich habe mal mit einer Person getestet, kurz vor'm macOS-Boot den Treiber der Karte zu entladen, in der Hoffnung, dass dieser Rückgängig macht, was auch immer den Apple-Treiber "stört" - hat nicht funktioniert, könnte aber unter 10.13 wiederholt werden.

m.M.n. ist der einzige vernünftige Weg, sollte 4) nicht greifen, dass man den Apple-Treiber zu Teilen in den Quellcode zurückübersetzt und dann Schritt für Schritt "mitgeht", um zu sehen, wann/wo genau der Treiber abbricht.